doi:10.1038/nature23461

Post-quantum cryptography

Daniel J. Bernstein[1] & Tanja Lange[2]

Cryptography is essential for the security of online communication, cars and implanted medical devices. However, many commonly used cryptosystems will be completely broken once large quantum computers exist. Post-quantum cryptography is cryptography under the assumption that the attacker has a large quantum computer; post-quantum cryptosystems strive to remain secure even in this scenario. This relatively young research area has seen some successes in identifying mathematical operations for which quantum algorithms offer little advantage in speed, and then building cryptographic systems around those. The central challenge in post-quantum cryptography is to meet demands for cryptographic usability and flexibility without sacrificing confidence.

Attackers are recording, and sometimes forging, vast volumes of human communication. Some of this communication is protected by cryptographic systems such as the Rivest–Shamir–Adleman (RSA) system and elliptic-curve cryptography (ECC), but if quantum computing scales as expected then it will break both RSA and ECC. We are in a race against time to deploy post-quantum cryptography before quantum computers arrive.

This Review surveys what cryptography does and the damage done by quantum computing. It is important to understand that research in cryptography is not monolithic: cryptography makes progress not only through designs of new proposals but also through cryptanalysts studying the security of proposals and weeding out weak proposals. We describe some of the main candidates for post-quantum cryptography and conclude with a look to the future.

Introduction to cryptography

When a user visits a website starting with https, the user's computer (laptop, desktop, mobile phone or other device) uses Transport Layer Security (TLS) to connect securely to the web server. TLS combines a sequence of cryptographic operations to ensure that no third party can understand what is being sent (confidentiality); that no third party can modify the messages without being detected (integrity); and that no third party can impersonate one of the communicating parties (authenticity).

Call the web server 'Alice' and the user's computer 'Bob'. What identifies Alice and Bob to each other is that they share a secret value, a 'key'. They use this key to perform 'symmetric' cryptographic operations. We will return later to the question of how they both know a secret key in the first place.

Suppose Alice has a message m to send to Bob: a web page or a file. Alice and Bob both know an encryption key k_{enc}. Alice applies a 'symmetric encryption algorithm' using this key k_{enc} to encrypt the message m, producing a ciphertext c, which Alice sends through the internet to Bob. Bob applies a matching symmetric decryption algorithm using the same key k_{enc} to decrypt the ciphertext c, obtaining the original message m.

Alice and Bob also both know an authentication key k_{auth}. Alice applies a 'message-authentication code' (MAC) using this key k_{auth} to the ciphertext c, producing an authentication tag, which Alice also sends through the internet to Bob, proving that she has access to the key. Bob verifies the computation by applying the same MAC using the same key.

Symmetric encryption ensures the confidentiality of data in https. It ensures that a spy cannot see the contents of a message. Authentication ensures authenticity and integrity of messages: it prevents a spy from modifying the message or substituting a different message, pretending to be Alice.

Several choices of symmetric encryption algorithms and MACs are offered by https. Some of the MACs are built from 'hash functions'. A hash function maps strings of arbitrary length to strings of some fixed length n: for example 256-bit strings. In this article, we consider only hash functions designed to make the following operations computationally hard: (1) given a value z in the image of h, find a pre-image, that is, a string m with $h(m) = z$; (2) given a string m and $h(m)$, find a second pre-image, that is, a string $m' \neq m$ with $h(m) = h(m')$; and (3) find a collision, that is, strings $m \neq m'$ with $h(m) = h(m')$. Hash functions provide compact fingerprints of messages; a small change to the message produces a completely different fingerprint. This property is used in constructions of MACs.

The description so far has left open how Alice and Bob arrived at having a shared symmetric key. This part of the https connection uses 'public-key cryptography'.

In public-key cryptography, each party has two keys: a public key and a private key. The private key is known only to the party, whereas the public key can be made public. Given Alice's public encryption key, anybody can encrypt a message to her, whereas only she is in possession of the matching private key that she uses to decrypt.

In the https scenario, Alice is a web server and Bob is a browser. Bob contacts Alice first to download the public key, then encrypts a one-time symmetric key to it, and finally uses the symmetric key in the rest of the communication for encryption and authentication as described above. Alice decrypts Bob's initial message to obtain the shared symmetric key and then also uses that for the rest of the communication.

Another option in https is to use another public-key function, 'key exchange', that is very close to what is described so far but uses different mathematical functions. Instead of Bob encrypting a symmetric key to Alice's public key, Bob and Alice both do computations that jointly generate a symmetric key.

There is still an important problem: how does Bob know that the public key provided by Alice really belongs to her? In the case of https, this is handled by browsers downloading and verifying 'certificates'. The cryptographic function used here is public-key 'signatures', which authenticate messages with public-key cryptography.

In a signature system, Sam applies a signature algorithm using his private signing key to a message m, producing a signature. Everybody can verify this signature by applying a verification algorithm to m using Sam's public signing key. Internally, these algorithms typically apply a hash function to the message, along with other mathematical operations involving the keys.

[1]Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois 60607-7045, USA. [2]Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, 5612 AZ Eindhoven, The Netherlands.

Like a MAC, a signature system ensures authenticity and integrity of the message, but there are two important differences. First, a signature can be verified by anybody using the public key, whereas an authentication tag is between the parties sharing k_{auth}. Second, only Sam has the private key used to produce a valid signature, whereas an authentication tag from Alice to Bob could have been computed by Alice or Bob.

In https, Sam is a well-known trusted party such as the Internet Security Research Group; everyone knows Sam's public key. The message signed by Sam is a certificate linking Alice's identity to Alice's public key. Bob verifies the signature from Sam and can then confidently use Alice's public key for encryption.

The devastating impact of Shor's algorithm

In the popular RSA public-key system[1], the public key is a product $N = pq$ of two secret prime numbers p and q. The security of RSA relies critically on the difficulty of finding the factors p, q of N. However, in 1994, Shor[2] introduced a fast quantum algorithm to find the prime factorization of any positive integer N.

There has been some research into analysing and optimizing the exact costs of Shor's algorithm: in particular, the number of quantum bits (qubits) required and the number of qubit operations required. For example, a variant of Shor's algorithm by Beauregard[3] uses $O(n^3\log n)$ operations on $2n + 3$ qubits if $N = pq$ fits into n bits. One can reduce the number of operations to $n^{2+o(1)}$, at some expense in the number of qubits, where $o(1)$ means something that converges to 0 as $n \to \infty$. One can also run many of those operations in parallel.

Internally, Shor's algorithm evaluates a periodic function on a superposition of all inputs within a wide range; applies a quantum Fourier transform to obtain an approximate superposition of periods of the function; and measures this superposition to find a random period. The periodic function is $e \mapsto a^e \bmod N$, where a is a random integer coprime to N, the arrow indicates 'maps to', and the notation 'mod N' means the remainder upon division by N. If N is not a power of a prime (an easy case to recognize), then a random period reveals a factor of N with probability high enough to be a security problem.

Shor introduced a similar algorithm to quickly find periods of the function $e, f \mapsto g^e h^f \bmod p$, revealing k such that $h = g^k \bmod p$. Replacing multiplication mod p with addition of points on an elliptic curve mod p breaks ECC, a popular alternative[4,5] to RSA.

These algorithms, when applied to widely deployed public-key sizes for RSA and ECC, require billions of operations on thousands of logical qubits. Fault-tolerant[6] attacks seem likely to require trillions of operations on millions of physical qubits. Perhaps quantum computing will encounter a fundamental obstacle that prevents it from ever scaling successfully to such sizes. However, no such obstacles have been identified. Prudent risk management requires defending against the possibility that these attacks will be successful.

Grover's algorithm

Many more cryptographic systems are affected by an algorithm that Grover[7] introduced in 1996. This algorithm is also the foundation for most, although not all, of the positive applications that have been identified for quantum computing.

Grover originally described his algorithm as searching an unordered database of size N using $\sqrt{N}$ quantum queries. This description begs the question of why the database creator did not simply put the database into order, allowing it to be searched using $O(\log N)$ queries. A closer look at the details of Grover's algorithm also raises difficult questions regarding the physical cost of quantum database queries.

It is better to describe Grover's algorithm as searching for roots of a function f: that is, searching for solutions x to the equation $f(x) = 0$. If one out of every N inputs is a root of f, then Grover's algorithm finds a root using only about $\sqrt{N}$ quantum evaluations of f on superpositions of inputs. If f can be evaluated quickly by a small circuit, then quantum evaluations of f do not use many qubit operations. This circuit condition often holds for the functions f that appear in cryptography.

The 'Advanced Encryption Standard' (AES)[8] is an example of a symmetric encryption algorithm. Assume that a user is known to have encrypted 128-bit plaintexts '7' and '8' under a secret 128-bit AES key k, producing a 256-bit ciphertext $c = (\mathrm{AES}_k(7), \mathrm{AES}_k(8))$ visible to the attacker. Define $f(x) = (\mathrm{AES}_x(7), \mathrm{AES}_x(8)) - c$. This function f can be evaluated quickly (about 20,000 bit operations) by a small circuit, and Grover's algorithm finds a root of f using only about 2^{64} quantum evaluations of f (overall[9] about 2^{86} 'T gates' and a similar number of 'Clifford gates' applied to about 3,000 qubits). Presumably this root is k: unless AES is deeply flawed, there will be at most a few pairs of distinct 128-bit keys x, k with collisions $(\mathrm{AES}_x(7), \mathrm{AES}_x(8)) = (\mathrm{AES}_k(7), \mathrm{AES}_k(8))$, and the user will not have selected one of those keys by chance.

Grover's speed-up from N to $\sqrt{N}$ is not as devastating as Shor's speed-up. Furthermore, each of Grover's $\sqrt{N}$ quantum evaluations must wait for the previous evaluation to finish. To quantify this issue, define T as the number of serial evaluations that can be carried out in the time available: for example, if the quantum computer can evaluate f in a nanosecond, and if the attacker is prepared to run a computation lasting for a year, then $T \approx 2^{55}$. If $\sqrt{N}$ exceeds T, then Grover's algorithm cannot use fewer than N/T evaluations spread across N/T^2 parallel quantum processors. This is a factor T better than pre-quantum techniques, but it is possible that this improvement will be wiped out by the overhead of qubit operations being more expensive than bit operations, making Grover's algorithm useless—even if scalable quantum computers are built and run Shor's algorithm successfully.

On the other hand, if qubit operations are small enough and fast enough, then Grover's algorithm will threaten many cryptographic systems that aim for 2^{128} security, such as 128-bit AES keys. We recommend simply switching to 256-bit AES keys: the extra costs are rarely noticeable. 'Information-theoretic' MACs such as GMAC and Poly1305 already protect against quantum computers without any modifications: their security analysis already assumes an attacker with unlimited computing power.

Post-quantum cryptography

Table 1 summarizes the effects of Shor's and Grover's algorithms on typical cryptosystems. The table gives the impression that the advent of quantum computers destroys public-key cryptography, leaving only symmetric cryptography (with larger key sizes). Fortunately, RSA and ECC are not the only public-key systems.

In the following five sections, we review details of five proposals that have solidly resisted every suggested attack. In particular, nobody has

Table 1 | Examples of widely deployed cryptographic systems and their conjectured security levels

Name	Function	Pre-quantum security level	Post-quantum security level
Symmetric cryptography			
AES-128[8]	Symmetric encryption	128	64 (Grover)
AES-256[8]	Symmetric encryption	256	128 (Grover)
Salsa20[58]	Symmetric encryption	256	128 (Grover)
GMAC[59]	MAC	128	128 (no impact)
Poly1305[60]	MAC	128	128 (no impact)
SHA-256[61]	Hash function	256	128 (Grover)
SHA3-256[62]	Hash function	256	128 (Grover)
Public-key cryptography			
RSA-3072[1]	Encryption	128	Broken (Shor)
RSA-3072[1]	Signature	128	Broken (Shor)
DH-3072[42]	Key exchange	128	Broken (Shor)
DSA-3072[63,64]	Signature	128	Broken (Shor)
256-bit ECDH[4–6]	Key exchange	128	Broken (Shor)
256-bit ECDSA[66,67]	Signature	128	Broken (Shor)

Security levels shown are against the best pre-quantum and post-quantum attacks known. Security level b means that the best attacks use approximately 2^b operations. This optimization ignores parallelization requirements; see text for discussion of the impact of such requirements. For hash functions, 'security' in this table refers to pre-image security.

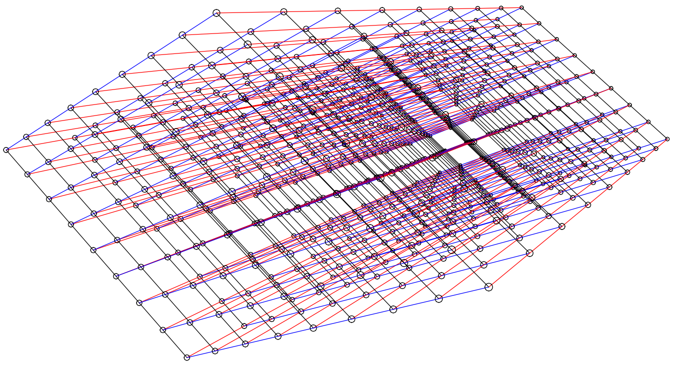

Figure 1 | Perspective view of a 9 × 9 × 9 subset of a non-orthogonal three-dimensional lattice. Lattice-based cryptography hides a point in a high-dimensional lattice mod q by making small changes to all coordinates. Code-based cryptography hides a point in a very-high-dimensional lattice mod 2 by changing some coordinates.

been able to figure out any useful way to apply Shor's algorithm and its generalizations to these proposals.

Choosing secure key sizes for these proposals does require attention to Grover's algorithm, along with generalizations of Grover's algorithm such as quantum walks. Simply doubling the target security level is adequate but generally imposes much more noticeable costs on public-key systems than on AES; these costs motivate research aimed at understanding the exact impact of Grover's algorithm, so as to be able to use smaller key sizes.

This is not a comprehensive list of proposals and attack ideas. We do not describe isogeny-based cryptography[10–12], for example, and we do not discuss Kuperberg's algorithm[13]. Our list is biased towards proposals that have survived decades of study.

Code-based encryption

High-reliability computer equipment uses an 'error-correcting code' to store 64 bits of logical data in 72 bits of physical memory. There is a 64×72 'generator matrix' G with each entry in the field $\mathbb{F}_2 = \{0, 1\}$ of integers modulo 2: in other words, a 64×72 matrix of bits. This matrix specifies each of the 72 physical bits as a sum, modulo 2, of some of the 64 logical bits. The code is $\mathbb{F}_2^{64}G$, a 64-dimensional subspace of the vector space $\mathbb{F}_2^{72}$, namely the subspace generated by the rows of G. The code is designed so that any single error in the 72 bits (any change of a bit to its opposite) can be reliably corrected, and any double error (changing any two bits) can be reliably detected.

Error-correcting codes can be scaled up to correct more errors in longer blocks of data. They are used in a wide range of applications, including hard drives, satellite communication, and fault-tolerant quantum computation.

In 1978, early in the history of public-key cryptography, McEliece[14] proposed using a generator matrix as a public key, and encrypting a codeword (an element of the code) by adding a specified number of errors to it. In formulas: the plaintext is a k-bit string m, the public key is a $k \times n$ matrix G, and the ciphertext is $mG + e$ where e is an n-bit string with w bits equal to 1. Examples of high-security parameters are $n = 6{,}960$, $k = 5{,}413$, and $w = 119$. The receiver can find e and m given $mG + e$ because the receiver secretly generates the code as a random 'Goppa code' that can efficiently correct w errors. This structure is not obvious from the generator matrix.

A simple but slow attack strategy against McEliece's system is 'information-set decoding' (ISD). An information set is a collection of codeword positions that determines the rest of the codeword. ISD guesses an information set, hoping that the ciphertext is error-free in those positions; uses linear algebra to (attempt to) determine the entire codeword; and checks that the ciphertext has the specified number of errors, in which case the codeword must be correct.

What makes ISD slow is that, for large matrices, the ciphertext is extremely unlikely to be error-free on any particular information set. More precisely, the number of guesses is $(c + o(1))^w$, where w is the number of errors added and $c > 1$ is a constant that depends on the selected ratio between the number of matrix rows and columns.

Dozens of attack papers against McEliece's system have found many improvements to ISD, but all of the pre-quantum attacks still take time $(c + o(1))^w$ for the same c, a remarkably stable track record. There have been some improvements within the $o(1)$ but these have not had much impact on security levels. McEliece's original key sizes (with $n = 1{,}024$, $k = 524$ and $w = 50$) were designed for 2^{64} security, and our successful attack against those key sizes 30 years later[15] took more than 2^{60} CPU cycles. All known algorithms to find the secret Goppa-code structure take even more time. The only post-quantum change in c has been a straightforward application[16] of Grover's algorithm, replacing c with $\sqrt{c}$.

Some modifications to McEliece's original system are important for improving security and performance. Rather than sending a message as a codeword, one should encrypt a random codeword, using a hash of the codeword as a secret key to authenticate and encrypt a message (see the description of https above); this protects McEliece's system against 'chosen-ciphertext attacks' in which an active attacker sees the results of decrypting modified ciphertexts. Another improvement to McEliece's system, due to Niederreiter[17], is to compress public keys to 'systematic form'. When k bits are encoded as n bits, 'systematic form' means that the first k physical bits are exactly the k logical bits, so the first $k \times k$ sub-matrix of the generator matrix is the identity matrix, which need not be transmitted. Yet another improvement, also due to Niederreiter, is to send 'syndromes' rather than erroneous codewords; this reduces the ciphertext size to about 200 bytes at a high security level.

The main practical problem with these systems is the key size, roughly a megabyte (in systematic form) at a high security level. Many newer code-based systems put more structure into public keys to allow more compression, but some of those proposals have been broken. The only post-quantum public-key-encryption systems that have received enough study for us to recommend are the original McEliece/Niederreiter systems.

Lattice-based encryption

In the 1990s, Hoffstein, Pipher and Silverman[18] introduced an encryption system, 'NTRU', that has much smaller keys than McEliece's system and that remains unbroken today. This system works as follows.

The public key is a p-coefficient polynomial $h = h_0 + h_1x + \cdots + h_{p-1}x^{p-1}$, with each coefficient in the set $\{0, 1, \ldots, q - 1\}$. A typical choice is $p = 743$ and $q = 2{,}048 = 2^{11}$; then the public key has $743 \times 11 = 8{,}173$ bits.

A ciphertext is another polynomial c in the same range. The sender chooses two secret polynomials d, e with small (say $-1, 0, 1$) coefficients, and computes $c = ((hd + e) \bmod x^p - 1) \bmod q$. The notation 'mod $x^p - 1$' means that x^p is replaced by 1, x^{p+1} is replaced by x, and so on.

Define L as the set of pairs (u, v) of p-coefficient polynomials with integer coefficients such that $0 = ((hu - v) \bmod x^p - 1) \bmod q$. Then L is a lattice in $2p$-dimensional space, and it contains a point close to $(0, c)$, namely $(d, c - e)$. The attacker's problem of finding the secrets d, e given the ciphertext c and public key h is thus an example of finding a lattice point close to a given point. This problem is analogous to the decoding problem for codes (see Fig. 1), except that here 'close' is interpreted as every coefficient being small, whereas codes simply count the number of non-zero coefficients.

NTRU, like McEliece's system, secretly generates the public key in a way that makes decoding efficient. Specifically, the receiver starts with a short vector of the form $(g, 3f)$, and uses a Euclidean algorithm to find h such that the lattice contains this vector, that is, such that $0 = ((hg - 3f) \bmod x^p - 1) \bmod q$. Then $(cg \bmod x^p - 1) \bmod q$ is the same as $((3df + eg) \bmod x^p - 1) \bmod q$. An analysis of coefficient sizes shows that $(3df + eg) \bmod x^p - 1$ almost certainly has all coefficients strictly between $-q/2$ and $q/2$, and then it is an easy exercise to find (d, e) given f and g.

There are many potential attack avenues against NTRU and other lattice-based cryptosystems (such as cryptosystems based on 'ring learning with errors'[19]). For example, very recently the 'cyclotomic' structure of x^p-1 has been used to break[20–22] some lattice-based cryptosystems by an extension of Shor's algorithm. NTRU is not known to be affected, but this attack avenue is new and has not been adequately explored. We recommend[23,24] replacing x^p-1 with x^p-x-1, avoiding this structure. As another example, recent attacks[25–27] that work for arbitrary lattices, without exploiting any polynomial structure, have smaller exponents than the best such attacks known just a few years ago; such lattices are used in cryptosystems based on 'learning with errors'[28]. Much more research is required to gain confidence in the security of lattice-based cryptography.

Lattice-based signatures

The first attempts[29–31] to turn hard lattice problems into signature systems were marred by attacks[32,33], and surviving systems suffered from large signature sizes. The most promising signature systems are based on Lyubashevsky's signature system[34] from 2012. Despite its evident youth, we decided to include it because the resulting signatures are relatively short and fast to compute.

The system is most easily presented using integer matrices. Implementations typically use polynomial rings and fast Fourier transforms for compact representations and efficiency. Lyubashevsky's system uses several system parameters, namely integers k, m, n determining the sizes of matrices, κ limiting the Hamming weight of certain vectors, and q a modulus. Let A be an $n\times m$ integer matrix modulo q, that is, $A\in\mathbb{Z}_q^{n\times m}$; this matrix may be shared by all users of the system but can also be chosen individually. The private key is a matrix $S\in\mathbb{Z}^{m\times k}$ with small entries, where 'small' means much smaller than q and is often restricted to $\{-1, 0, 1\}$. The public key is the $n\times k$ matrix $T=AS$, where the entries are computed modulo q. If A is not shared then it is also part of the public key.

The system uses a hash function $H:\mathbb{Z}_q^n\times\{-1,0,1\}^*\rightarrow\{0,1\}^k$, where the output vectors additionally satisfy that no more than κ entries are nonzero. It is easy to build H from a traditional hash function h by encoding inputs and outputs appropriately.

The signer starts by picking y from an m-dimensional distribution, typically a discrete Gaussian distribution. (A discrete Gaussian distribution is a distribution obtained by considering only integer values of the regular Gaussian distribution and normalizing appropriately.) The signer then computes $c=H(Ay \bmod q, \mu)$, where μ is the message, and $z=Sc+y$. The signature is the pair (c, z). To avoid leaking information about the private key S through the distribution of (c, z), Lyubashevsky uses 'rejection sampling' to force an S-independent distribution. This means that the process is restarted with probability depending on (c, z).

The signature is accepted as valid if c and z are sufficiently small and if $H(Az-Tc \bmod q, \mu)=c$. The latter holds for valid signatures because $Az-Tc\equiv A(Sc+y)-ASc\equiv Ay \bmod q$.

Later proposals such as BLISS[35] improve the running time by reducing the frequency of rejection in the last step. A ring version with $k=n$ and $m=2n$ signs in under half a millisecond and verifies about 10 times faster. Public keys and signatures each are between 5 and 7 kilobits, not much larger than RSA signatures.

Ongoing challenges include (1) generating the distribution in a way that does not leak[36] information on S through 'side channels' (see below) and (2) analysing the security of the underlying algorithmic problem, namely the problem of finding short integer solutions to a system of equations modulo q.

Multivariate-quadratic-equation signatures

Matsumoto and Imai[37] introduced a new signature system, 'C*', in 1988. Patarin[38] broke the C* system in 1995 but the next year[39] introduced a stronger system, 'HFE^{v-}', that remains unbroken today.

The HFE^{v-} public key is a sequence of polynomials $p_1, \ldots, p_m$ in the n-variable polynomial ring $\mathbb{F}_2[x_1, \ldots, x_n]$ over the field $\mathbb{F}_2$, with $m\le n$. The polynomials are limited to quadratics and have no squared terms: each polynomial p_i has the form $a_i+\sum_j b_{i,j}x_j+\sum_{j<k}c_{i,j,k}x_jx_k$ with $a_i, b_{i,j}, c_{i,j,k}\in\mathbb{F}_2$. The coefficients have no obvious public structure.

A signature of a message is an n-bit string $(s_1, \ldots, s_n)\in\mathbb{F}_2^n$ such that the m-bit string $(p_1(s_1, \ldots, s_n), \ldots, p_m(s_1, \ldots, s_n))\in\mathbb{F}_2^m$ equals a standard m-bit hash $(h_1, \ldots, h_m)$ of the message. An example of a reasonable parameter choice (including the internal parameters v, q described below) is $(m, n, v, q)=(240, 272, 16, 2^{256})$; then a signature is just 34 bytes. These very short signatures are an attractive feature of this signature system.

The signer chooses the polynomials with a secret structure that allows the signer to solve the simultaneous quadratic equations $p_1(s_1, \ldots, s_n)=h_1, \ldots, p_m(s_1, \ldots, s_n)=h_m$. Specifically, HFE^{v-} exploits the fact that there are general methods to solve polynomial equations of degree d over finite fields $\mathbb{F}_q$ in time $(d\log q)^{O(1)}$ if the equations are in just one variable. We now explain how the multivariate polynomials $p_1, \ldots, p_m$ are secretly related to a univariate polynomial.

The signer views an n-bit signature $(s_1, \ldots, s_n)$ as a randomly chosen v-bit string $(r_1, \ldots, r_v)\in\mathbb{F}_2^v$, where $v\le n-m$, together with an $(n-v)$-bit element $S\in\mathbb{F}_q$, where $q=2^{n-v}$. This view is secret: v and q can be standardized, but before the signature is partitioned into (r, S) it is passed through a secret invertible $n\times n$ matrix chosen by the signer. This means that S is some linear function of $s_1, \ldots, s_n$, but not a public linear function.

The signer similarly views an m-bit hash value, together with a randomly chosen $(n-v-m)$-bit string, as an element $H\in\mathbb{F}_q$. This view is also not standardized: it is obscured by another secret matrix. Here is the overall signing process, starting from the hash value: choose the $v+(n-v-m)=n-m$ random bits mentioned above; construct H; try to solve for S as explained below (or, if no solution S exists, start over with another choice of random bits); and construct the resulting n-bit signature.

There is one more secret: a degree-d polynomial $P\in\mathbb{F}_q[x, y_1, \ldots, y_v]$ of the form $A+\sum_j B_jx^{2^j}+\sum_{j>k}C_{j,k}x^{2^j+2^k}+\sum_j D_jy_j+\sum_{j,k}E_{j,k}y_jx^{2^k}+\sum_{j>k}F_{j,k}y_jy_k$. This polynomial specifies a secret equation connecting S and H, namely $P(S, r_1, \ldots, r_v)=H$. To convert this equation into the public quadratic polynomials, the signer writes each bit of S^{2^j} as a linear combination of $s_1, \ldots, s_n$. To solve the equation for any particular signature, the signer simply observes that this is a univariate equation in S for any particular choice of random bits $r_1, \ldots, r_v$.

For comparison, C* takes $q=2^m=2^n$; takes the polynomial P as a monomial x^{2^j+1} with exponent coprime to 2^q-1; and solves the equation $S^{2^j+1}=H$ by computing $S=H^e$, where e is the reciprocal of $2^j+1 \bmod 2^q-1$. The core idea of Patarin's attack is that the bilinear equation $S^{2^{2j}}H=H^{2^j}S$ is equivalent to a secret bilinear equation E on the bits of hashes and signatures. Each hash–signature pair produces a linear equation for the secret coefficients of E, and after enough signatures the attacker simply solves for those coefficients, at which point signature forgery is easy. HFE^{v-} blocks this attack by including more monomials in P.

There is a vast literature on other multivariate-quadratic signature systems and on algorithms to attack these systems. For HFE^{v-} in

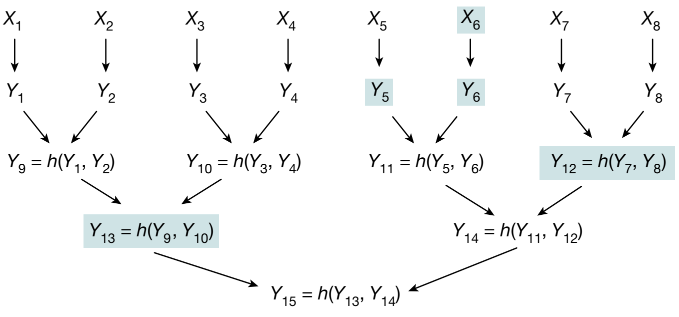

Figure 2 | Merkle tree with public key Y_{15} to sign eight messages. The boxes highlight the values constituting the sixth signature. This signature reveals parts of X_6, the secret key of Lamport's one-time signature. It also includes the matching Lamport public key Y_6, along with Y_5, Y_{12} and Y_{13}. The verifier computes Y_{15} and checks it against the Merkle public key.

Table 2 | Qualitative overview of the described post-quantum systems

Approach	Advantages	Disadvantages
Code-based encryption (using Goppa codes)	High confidence in security; very fast encryption; short ciphertexts	Large public keys
Lattice-based encryption (using NTRU or related)	Short ciphertexts and keys; very fast encryption	Require more security analysis
Lattice-based signatures	Short keys and signatures; fast	Require more security analysis; side-channel attacks on discrete Gaussians
Multivariate-quadratic-equation signatures	Very short signatures	Require more security analysis
Hash-based signatures (stateful version)	High confidence; simple description	Management of state
Hash-based signatures (stateless version)	High confidence; simple description	Large signatures

particular, all known attacks take time exponential in approximately $(n-m+\lceil \log_2 d\rceil)\log_2 n$, where the ceiling function $\lceil x\rceil$ denotes the smallest integer greater than or equal to the real number x. The same type of analysis used for recent pre-quantum parameter choices[40] indicates that $m=240$, $n-v=256$ and $n=272$ provide high post-quantum security against known attacks even if d is quite small.

Hash-based signatures

One of the design goals mentioned earlier for hash functions is that finding a pre-image for a given output string is computationally hard. In 1975, Lamport realized that this could be used to build a one-time signature system[41,42].

Lamport's one-time signatures work as follows. To generate his key pair, Sam chooses two random strings x_0 and x_1; these constitute his secret key. His public key is $(h(x_0), h(x_1))$, where h is a hash function which is known to everybody. If he wants to sign 0 he reveals x_0; the verifier recomputes $h(x_0)$ and checks the result against the first half of the public key. To sign 1, Sam reveals x_1. More generally, to sign an m-bit message, Sam takes $2m$ strings as the secret key $X=(x_{10}, x_{11}, x_{20}, x_{21}, \ldots, x_{m0}, x_{m1})$ and their hash values as public key $Y=(h(x_{10}), h(x_{11}), h(x_{20}), h(x_{21}), \ldots, h(x_{m0}), h(x_{m1}))$. The signature of, for example, 10110… is $(x_{11}, x_{20}, x_{31}, x_{41}, x_{50}, \ldots)$. Security rapidly degrades if Sam signs more than one message under the same key.

To overcome the problems of large public keys that could be used only once, Merkle proposed[43,44] to combine 2^k public keys into one which can then be used to verify all 2^k signatures. For that, create 2^k key pairs for Lamport's one-time signature and arrange the public keys $Y_1, Y_2, \ldots, Y_{2^k}$ as the leaves of a binary tree with $k+1$ levels. A binary tree is one in which each node has exactly three edges, two going to a level closer to the leaves and one going closer to the root, except for the leaf nodes having only one and the root node having only two edges. Figure 2 shows an example of a Merkle tree with $2^3=8$ leaves. To compute the public key combining these 2^k keys, start from the leaves and compute the hash of each pair of public keys connected by edges in the tree, starting with $Y_{2^k+1}=h(Y_1, Y_2)$; continue iteratively through the levels, ending by computing the root $Y_{2^{k+1}-1}=h(Y_{2^{k+1}-3}, Y_{2^{k+1}-2})$. The value $Y_{2^{k+1}-1}$ at the root node is the public key of the system.

The public key is now a single hash value, but the signatures need to include more information to make it possible to check them. As before, a signature using secret key X_i reveals the x_{ij} matching the bit pattern of the message to be signed; in addition, the matching public key Y_i is included so that the Lamport signature can be verified. The signature also includes all siblings to the nodes encountered on the path from Y_i to the root; signature verification links Y_i to the public key by computing all hash values towards the root and comparing the value at the root with the public key.

Hash functions appear in all signature systems. Standard hash functions are affected only by Grover's attack, not by Shor's attack. This makes Merkle's very simple signatures excellent candidates for post-quantum signatures: they have a clear security track record, and computing hash functions is very fast.

Various improvements exist: using better one-time signatures[45,46] to decrease the signature size, for example, or building trees of trees to reduce key-generation time. A system based on XMSS[47,48] is currently in the final steps of adoption for internet protocols by the Internet Research Task Force (IRTF). The US National Institute for Standards and Technology (NIST) has indicated that they will fast-track a hash-based signature system.

It is important to never reuse a secret key X_i: each X_i is usable only one time. This means that the system described so far is 'stateful': the signer needs to remember which keys have been used. This might sound easy but has been described as a "huge foot-cannon"[49]: it poses problems for environments that use, for example, virtual machines or shared signing keys. For such applications, 'stateless' systems exist[50], but this feature comes at the expense of longer signatures and longer signature-generation time.

Integration into the real world

Deploying a cryptographic system incurs physical costs: the time and energy consumed by cryptographic computations and by communication of keys, signatures, and so forth. Today's deployment of cryptography for billions of users relies on the fact that cryptography fits the users' budget. For comparison, some of the simplest goals of cryptography might also be achieved by couriers transporting locked briefcases, but this is so expensive that very few users can afford it.

Deploying a cryptographic system also raises questions of whether the real world matches the system's mathematical models of user capabilities and attacker capabilities. The most important examples are 'side-channel attacks', in which the attacker learns extra information by observing physical effects such as timing[51] or power consumption[52]. Another example is the problem of statefulness mentioned above.

A large part of cryptographic research is aimed at finding the maximum real-world security achievable under various constraints on real-world costs. For example, side-channel attacks against cryptography are the largest topic at the immensely popular 'Cryptographic Hardware and Embedded Systems' conference series, whereas there seems to have been far less public analysis of, for example, the power of side-channel attacks against locked briefcases. As a final example, a state-of-the-art implementation[53] of McEliece's code-based system takes even less processing time than ECC; the only serious obstacle to wide deployment of this system is its key size.

Standardization

Several standardization bodies have recognized the urgency of switching to cryptosystems that remain secure against attacks by quantum computers. This is an important development because many applications of cryptography require all parties to use the same cryptographic system: standardization is thus a prerequisite for widespread deployment. Sometimes *de facto* standards are set without standardization bodies, but formal standardization processes are widely viewed as reducing cryptographic risks.

The Internet Engineering Task Force (IETF) and its research branch IRTF are leading with having almost finalized standardization of a hash-based signature system. NIST has opened a call for submissions of candidates for standardization; the submission deadline is November 2017, and evaluation is expected to run for 3–5 years. This call should result in the recommendation of a small portfolio of systems for encryption, signatures and possibly other key-exchange mechanisms. Other standardization bodies with post-quantum cryptography on the agenda are ETSI, with their 'quantum-safe' working group; ISO, with SC27 WG2; and OASIS, with the KMIP standard.

One of the big European players in post-quantum cryptography is the EU Horizon 2020 PQCRYPTO project. The logo of the project is a Galapagos tortoise, illustrating the state of post-quantum cryptography at the start of that project: confidence-inspiring proposals of long-lived systems are too big or too slow for casual deployment. This project has already released recommendations[54] of such confidence-inspiring systems for users who can afford them, such as AES-256, McEliece's code-based cryptosystem and hash-based signatures.

Even without standardization, there are already experiments with wide-scale deployment of post-quantum cryptography. A notable example is Google's recent experiment[55] with the recent 'New Hope'[56] lattice-based cryptosystem: an update to the Google Chrome browser automatically encrypted data with New Hope (and with ECC) for a fraction of all Chrome users connecting to Google websites. This experiment concluded[57] that there were "no reported problems" and that quick deployment of this cryptosystem is practical. Whether the cryptosystem is secure is a different question.

Ongoing and future work

These are exciting times for post-quantum cryptography. Researchers have identified many different ways to provide critical functions such as public-key encryption and public-key signatures: see Table 2 for illustrative examples. Some of these proposals have survived many years of scrutiny, but these proposals incur serious costs, especially in network traffic. Other proposals are more attractive for deployment, but their security is less clear, and it is likely that some of those proposals will be broken.

We expect that most of the systems sketched in this review will stand the test of time but probably with different parameters. We expect an increased uptake in research in post-quantum cryptography motivated by NIST's competition—more designs, more optimizations and implementations, and also more attacks. An important part of progress in cryptography is to understand what not to do—that is, what systems are vulnerable to attacks. Only once systems are sufficiently well studied to be considered secure does it make sense to establish practicality. Much more work is needed to build post-quantum systems that are widely deployable while at the same time inspiring confidence.

Received 28 February; accepted 30 June 2017.

1. Rivest, R. L., Shamir, A. & Adleman, L. M. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21,** 120–126 (1978)
2. Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Ann. Symp. on Foundations of Computer Science (FOCS '94)* 124–134 (IEEE, 1994).
3. Beauregard, S. Circuit for Shor's algorithm using $2n + 3$ qubits. *Quantum Inf. Comput.* **3,** 175–185 (2003).
4. Miller, V. S. Use of elliptic curves in cryptography. In *Advances in Cryptology, Proc. CRYPTO '85* (ed. Williams, H. C.) 417–426 (Springer, 1985).
5. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **48,** 203–209 (1987).
6. Campbell, E. T., Terhal, B. M. & Vuillot, C. Roads towards fault-tolerant universal quantum computation. *Nature* http://dx.doi.org/10.1038/nature23460 (2017).
7. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proc. 28th Ann. ACM Symp. on Theory of Computing* (ed. Miller, G. L.) 212–219 (ACM, 1996).
8. Daemen, J. & Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard* (Springer, 2002).
9. Grassl, M., Langenberg, B., Roetteler, M. & Steinwandt, R. Applying Grover's algorithm to AES: quantum resource estimates. In *Post-Quantum Cryptography, Proc. 7th International Workshop (PQCRYPTO 2016)* (ed. Takagi, T.) 29–43 (Springer, 2016).
10. Rostovtsev, A. & Stolbunov, A. Public-key cryptosystem based on isogenies. Preprint at https://eprint.iacr.org/2006/145 (2006).
11. Couveignes, J.-M. Hard homogeneous spaces (2006). Preprint at https://eprint.iacr.org/2006/291.
12. Jao, D. & de Feo, L. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography, Proc. 4th International Workshop (PQCRYPTO 2011)* (ed. Yang, B.-Y.) 19–34 (Springer, 2011).
13. Kuperberg, G. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.* **35,** 170–188 (2005).
14. McEliece, R. J. *A Public-Key Cryptosystem based on Algebraic Coding Theory*. Deep Space Network Progress Report 42–44 http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF (1978).
15. Bernstein, D. J., Lange, T. & Peters, C. Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography, Proc. 2nd International Workshop (PQCRYPTO 2008)* (eds Buchmann, J. A. & Ding, J.) 31–46 (Springer, 2008).
16. Bernstein, D. J. Grover vs. McEliece. In *Post-Quantum Cryptography, Proc. 3rd International Workshop (PQCRYPTO 2010)* (ed. Sendrier, N.) 73–80 (Springer, 2010).
17. Niederreiter, H. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control Inform.* **15,** 159–166 (1986).
18. Hoffstein, J., Pipher, J. & Silverman, J. H. NTRU: a ring-based public key cryptosystem. In *Algorithmic Number Theory, Proc. 3rd International Symp. (ANTS-III)* (ed. Buhler, J.) 267–288 (Springer, 1998).
19. Lyubashevsky, V., Peikert, C. & Regev, O. On ideal lattices and learning with errors over rings. *J. ACM* **60, 43**:1–43:35 (2013).
20. Campbell, P., Groves, M. & Shepherd, D. Soliloquy: a cautionary tale. http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf (2014).
21. Biasse, J.-F. & Song, F. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *Proc. 27th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2016)* (ed. Krauthgamer, R.) 893–902 (SIAM, 2016).
An extension of Shor's algorithm breaks some lattice-based systems.
22. Cramer, R., Ducas, L. & Wesolowski, B. Short Stickelberger class relations and application to Ideal-SVP. In *Advances in Cryptology, Proc. Ann. International Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2017)* 324–348 (Springer, 2017).
23. Bernstein, D. J. A subfield-logarithm attack against ideal lattices. The cr.yp.to blog https://blog.cr.yp.to/20140213-ideal.html (2014).
24. Bernstein, D. J., Chuengsatiansup, C., Lange, T. & van Vredendaal, C. NTRU Prime. Preprint at https://eprint.iacr.org/2016/461 (2016).
25. Laarhoven, T. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *Advances in Cryptology, Proc. 35th Ann. Cryptology Conf. (CRYPTO 2015)* (eds Gennaro, R. & Robshaw, M.) 3–22 (Springer, 2015).
26. Laarhoven, T. & de Weger, B. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In *Progress in Cryptology, Proc. 4th International Conf. on Cryptology and Information Security in Latin America (LATINCRYPT 2015)* (eds Lauter, K. E. & Rodríguez-Henríquez, F.) 101–118 (Springer, 2015).
27. Becker, A., Ducas, L., Gama, N. & Laarhoven, T. New directions in nearest neighbor searching with applications to lattice sieving. In *Proc. 27th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2016)* (ed. Krauthgamer, R.) 10–24 (SIAM, 2016).
28. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56, 34**:1–34:40 (2009).
29. Goldreich, O., Goldwasser, S. & Halevi, S. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology, Proc. 17th Ann. International Cryptology Conf. (CRYPTO'97)* (ed. Kaliski, B. S. Jr) 112–131 (Springer, 1997).
30. Hoffstein, J., Pipher, J. & Silverman, J. H. NSS: an NTRU lattice-based signature scheme. In *Advances in Cryptology, Proc. International Conf. on the Theory and Application of Cryptographic Techniques (EUROCRYPT 2001)* (ed. Pfitzmann, B.) 211–228 (Springer, 2001).
31. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J. H. & Whyte, W. NTRUSIGN: digital signatures using the NTRU lattice. In *Topics in Cryptology, Proc. Cryptographers' Track at the RSA Conf. 2003 (CT-RSA 2003)* (ed. Joye, M.) 122–140 (Springer, 2003).
32. Nguyen, P. Q. & Regev, O. Learning a parallelepiped: cryptanalysis of GGH and NTRU signatures. In *Advances in Cryptology, Proc. 25th Ann. International Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2006)* (ed. Vaudenay, S.) 271–288 (Springer, 2006).
33. Ducas, L. & Nguyen, P. Q. Learning a zonotope and more: cryptanalysis of NTRUSign countermeasures. In *Advances in Cryptology, Proc. 18th International Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012)* (eds Wang, X. & Sako, K.) 433–450 (Springer, 2012).
34. Lyubashevsky, V. Lattice signatures without trapdoors. In *Advances in Cryptology, Proc. 31st Ann. International Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012)* (eds Pointcheval, D. & Johansson, T.) 738–755 (Springer, 2012).
35. Ducas, L., Durmus, A., Lepoint, T. & Lyubashevsky, V. Lattice signatures and bimodal Gaussians. In *Advances in Cryptology, Proc. 33rd Ann. Cryptology Conf. (CRYPTO 2013)* (eds Canetti, R. & Garay, J. A.) 40–56 (Springer, 2013).
36. Groot Bruinderink, L., Hülsing, A., Lange, T. & Yarom, Y. Flush, Gauss, and reload: a cache attack on the BLISS lattice-based signature scheme. In *Cryptographic Hardware and Embedded Systems, Proc. 18th International Conf. (CHES 2016)* (eds Gierlichs, B. & Poschmann, A. Y.) 323–345 (Springer, 2016).
First successful side-channel attacks against lattice-based signatures.
37. Matsumoto, T. & Imai, H. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology, Proc. Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'88)* (ed. Günther, C. G.) 419–453 (Springer, 1988).
38. Patarin, J. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In *Advances in Cryptology, Proc. 15th Ann. International Cryptology Conf. (CRYPTO'95)* (ed. Coppersmith, D.) 248–261 (Springer, 1995).

39. Patarin, J. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology, Proc. International Conf. on the Theory and Application of Cryptographic Techniques (EUROCRYPT'96)* (ed. Maurer, U. M.) 33–48 (Springer, 1996).
40. Petzoldt, A., Chen, M.-S., Yang, B.-Y., Tao, C. & Ding, J. Design principles for HFEv-based multivariate signature schemes. In *Advances in Cryptology, Proc. 21st International Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2015)* (eds Iwata, T. & Cheon, J. H.) 311–334 (Springer, 2015).
Optimizes conservative multivariate-quadratic signatures.
41. Lamport, L. *Constructing Digital Signatures from a One Way Function*. Technical Report No. SRI-CSL-98 (SRI International Computer Science Laboratory, 1979); available at http://lamport.azurewebsites.net/pubs/pubs.html#dig-sig
42. Diffie, W. & Hellman, M. E. New directions in cryptography. *IEEE Trans. Inf. Theory* **22,** 644–654 (1976).
43. Merkle, R. C. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Stanford Univ., http://www.merkle.com/papers/Thesis1979.pdf (1979).
44. Merkle, R. C. A certified digital signature. In *Advances in Cryptology, Proc. 9th Ann. International Cryptology Conf. (CRYPTO '89)* (ed. Brassard, G.) 218–238 (Springer, 1989).
45. Dods, C., Smart, N. P. & Stam, M. Hash based digital signature schemes. In *Cryptography and Coding, Proc. 10th IMA International Conf.* (ed. Smart, N. P.) 96–115 (Springer, 2005).
46. Hülsing, A. W-OTS$^+$—shorter signatures for hash-based signature schemes. In *Progress in Cryptology, Proc. 6th International Conf. on Cryptology in Africa (AFRICACRYPT 2013)* (eds Youssef, A., Nitaj, A. & Hassanien, A. E.) 173–188 (Springer, 2013).
47. Buchmann, J. A., Dahmen, E. & Hülsing, A. XMSS—a practical forward secure signature scheme based on minimal security assumptions. In *Post-Quantum Cryptography, Proc. 4th International Workshop (PQCRYPTO 2011)* (ed. Yang, B.-Y.) 117–129 (Springer, 2011).
Conservative stateful hash-based signatures are small and fast.
48. Hülsing, A., Rausch, L. & Buchmann, J. A. Optimal parameters for XMSS^{MT}. In *Security Engineering and Intelligence Informatics, Proc. CD-ARES 2013 Workshops: MoCrySEn and SeCIHD* (eds Cuzzocrea, A. *et al.*) 194–208 (Springer, 2013).
49. Langley, A. Hash based signatures. *Imperial Violet* https://www.imperialviolet.org/2013/07/18/hashsig.html (2013).
50. Bernstein, D. J. *et al.* SPHINCS: practical stateless hash-based signatures. *In Advances in Cryptology, Proc. 34th Ann. International Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015)* (eds Oswald, E. & Fischlin, M.) 368–397 (Springer, 2015).
Conservative stateless hash-based signatures are practical.
51. Kocher, P. C. Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In *Advances in Cryptology, Proc. 16th Ann. International Cryptology Conf. (CRYPTO '96)* (ed. Koblitz, N.) 104–113 (Springer, 1996).
52. Kocher, P. C., Jaffe, J. & Jun, B. Differential power analysis. In *Advances in Cryptology, Proc. 19th Ann. International Cryptology Conf. (CRYPTO '99)* (ed. Wiener, M. J.) 388–397 (Springer, 1999).
53. Bernstein, D. J., Chou, T. & Schwabe, P. McBits: fast constant-time code-based cryptography. In *Cryptographic Hardware and Embedded Systems, Proc. 15th International Workshop (CHES 2013)* (eds Bertoni, G. & Coron, J.-S.) 250–272 (Springer, 2013).
Conservative code-based encryption is faster than ECC.
54. PQCRYPTO Project. Initial Recommendations of Long-Term Secure Post-Quantum Systems. https://pqcrypto.eu.org/docs/initial-recommendations.pdf (2015).
55. Braithwaite, M. Experimenting with post-quantum cryptography. *Google Security Blog*. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html (2016).
56. Alkim, E., Ducas, L., Pöppelmann, T. & Schwabe, P. Post-quantum key exchange—a new hope. In *25th USENIX Security Symp. (USENIX Security 16)* (eds Holz, T. & Savage, S.) 327–343 (USENIX Association, 2016).
57. Langley, A. CECPQ1 results. *Imperial Violet* https://www.imperialviolet.org/2016/11/28/cecpq1.html (2016).
58. Bernstein, D. J. The Salsa20 family of stream ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists* (eds Robshaw, M. J. B. & Billet, O.) 84–97 (Springer, 2008).
59. McGrew, D. A. & Viega, J. The security and performance of the Galois/counter mode (GCM) of operation. In *Progress in Cryptology, Proc. 5th International Conf. on Cryptology in India (INDOCRYPT 2004)* (eds Canteaut, A. & Viswanathan, K.) 343–355 (Springer, 2004).
60. Bernstein, D. J. The Poly1305-AES message-authentication code. In *Fast Software Encryption, Proc. 12th International Workshop (FSE 2005)* (eds Gilbert, H. & Handschuh, H.) 32–49 (Springer, 2005).
61. NIST Information Technology Laboratory. *Secure Hash Standard (SHS).* Federal Information Processing Standards Publication 180-4, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180–4.pdf (NIST, 2012).
62. Bertoni, G., Daemen, J., Peeters, M. & Assche, G. V. Keccak. In *Advances in Cryptology, Proc. 32nd Ann. International Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2013)* (eds Johansson, T. & Nguyen, P. Q.) 313–314 (Springer, 2013).
63. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proc. CRYPTO '84* (eds Blakley, G. R. & Chaum, D.) 10–18 (Springer, 1984).
64. Schnorr, C.-P. Efficient identification and signatures for smart cards. In *Advances in Cryptology, Proc. 9th Ann. International Cryptology Conf. (CRYPTO '89)* (ed. Brassard, G.) 239–252 (Springer, 1989).
65. Bernstein, D. J. Curve25519: new Diffie–Hellman speed records. In *Public Key Cryptography, Proc. 9th International Conf. on Theory and Practice of Public-Key Cryptography (PKC 2006)* (eds Yung, M. *et al.*) 207–228 (Springer, 2006).
66. Johnson, D., Menezes, A. & Vanstone, S. A. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.* **1,** 36–63 (2001).
67. Bernstein, D. J., Duif, N., Lange, T., Schwabe, P. & Yang, B.-Y. High-speed high-security signatures. *J. Cryptographic Eng.* **2,** 77–89 (2012).

Acknowledgements We thank A. Hülsing and B.-Y. Yang for their comments. Author list is in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf. This work was supported by the European Commission under Contract ICT-645622 PQCRYPTO; by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005; and by the US National Science Foundation under grant 1314919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (or other funding agencies).

Author Contributions D.J.B. and T.L. jointly inventoried the space of cryptographic systems, selected specific systems and quantum algorithms to cover, decided on the organization, and wrote text. No new experiments were performed.

Author Information Reprints and permissions information is available at www.nature.com/reprints. The authors declare no competing financial interests. Readers are welcome to comment on the online version of the paper. Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. Correspondence should be addressed to T.L. (tanja@hyperelliptic.org).